Maintaining and Comparing Requirements

Debbie Richards and Didar Zowghi
Department of Computing
Division of Information and Communications Sciences
Macquarie University
Sydney, Australia
Email: richards, didar@ics.mq.edu.au

Abstract: This paper offers a framework and a process model for comparing requirements from different stakeholders. The framework combines a knowledge based system approach to requirements elicitation, a mathematically based technique for conceptual modelling, a four-state model of comparison to identify different types of conflict, a number of strategies to resolve inconsistencies and a distance metric for determining if viewpoints are converging. The framework is an extension of a knowledge engineering framework for reconciling differences between multiple sources of expertise. To justify this extension a comparison is made between the nature of knowledge and requirements and the issues that face both disciplines. The paper focuses on the common issues of maintenance and combining requirements, which are regarded as a specialised type of knowledge.

Keywords: requirements engineering, knowledge engineering, ripple down rules, formal concept analysis

1 Using Knowledge Engineering Techniques for Requirements Engineering

A requirement is essentially knowledge about a need which must be elicited, specified and validated. Since system requirements are a type of knowledge, it is natural that researchers would look at applying knowledge engineering (KE) techniques to the problem of requirements engineering (RE). There is also similarity between the sources and methods of acquiring knowledge and requirements. For example, both may use experts, interviews and documentation as the source of development of a domain model for the proposed system. While KE research offers some possible approaches to the RE problem, it is important to note that KE has a number of problems of its own which must be considered as they impact on RE. These issues include [17]:

- the knowledge acquisition (KA) bottleneck (this refers to the difficulty of extracting knowledge from an expert since much expert action is subconscious and not easily described,
- the brittleness problem (the advice or explanation of a Knowledge Based System (KBS) tends to be unreliable, if not absurd, when asked a question outside of its current knowledge boundaries , and
- the difficulty associated with maintaining large KBS.

The first and last problems are of most relevance when we consider RE. To some extent the acquisition bottleneck is even greater when dealing with requirements because we are asking the user to envisage a possible system in addition to using knowledge they have of an existing system [6] and the user may not be an expert. In the RE community the maintenance issue is addressed by research into Requirements Volatility [21, 22, 46]. The RE framework provided in this paper is based on a KA technique, known as Ripple Down Rules (RDR) [4] that were developed to address the maintenance of large KBS issue. RDR also offers a proven solution to the KA bottleneck problem [10] and to a lesser extent the brittleness problem [11].

Another problem shared by KE and RE is the need to handle conflicting viewpoints[1] which results from using multiple sources. It appears that the RE community have been braver in acknowledging this problem and in seeking to address it. There is still no definitive solution but a number of alternatives are offered including a viewpoints approach [9, 12] and the lazy 'consistency' approach [28]. Through the recent incorporation of Formal Concept Analysis (FCA) [45], RDR also provides a way of detecting conflict between multiple KBS [34].

Greenspan, Mylopoulos and Borgida [15] have found requirements engineering to be essentially a modelling task. We take a similar view but emphasize that there are real problems in capturing, describing and validating models which by their very definition are only representations [2]. Mainstream KBS research focuses on the development of "good" models by the knowledge engineer before KA can commence. We prefer to acquire a simple model, that involves minimal analysis, directly from the expert using cases. The simple model is used to automatically uncover a concept lattice which represents a conceptual hierarchy of the knowledge of each expert. The user is presented with both sentential and diagrammatic representations of their models. The concepts derived from each set of specifications are compared according to the four quadrant model of comparison [38] which classifies two conceptual models as being in one of four states: *consensus*, *correspondence*, *conflict* or *contrast*. Negotiation strategies are offered to reconcile concepts not in a state of consensus with the goal being the development of a shared viewpoint.

This paper offers an RE framework which is built using the process model presented in Section 3. As RDR and FCA are major components of the framework they are briefly introduced in the next section. Section 4 considers future work and provides some concluding remarks.

2. The Theories

The framework offered in this paper is based on the use of the KA technique - Ripple Down Rules and the conceptual modelling technique - Formal Concept Analysis. To give the reader some familiarity with these methods the key ideas behind both theories are described.

2.1 Ripple Down Rules

RDR is a case-based reasoning approach to KA which stores knowledge in an exception structure as shown in Figure 1. Cases are used to prompt the acquisition of knowledge and are used to guide the expert in defining rules and to provide automatic validation of knowledge [5]. The RDR approach to KA is to perform an inference on a case and show the user the system-assigned conclusion/s. If the user agrees with the conclusions given they process the next case. If they do not agree with a conclusion they take the option to reclassify the case. Reclassification involves specifying the correct conclusion and picking some features in the case that justify the new conclusion. These features form the conditions of the new rule. The new rule is added as an exception to the rule that gave the misclassification. The case that prompts a rule to be added is stored in association with the new rule. When a new rule is added, the rule must distinguish between the present case and all the stored cases that can reach that rule. To do this, the expert is required firstly to construct a rule which distinguishes between the new case and one of the stored cases. If other stored cases satisfy the rule, further conditions are required to be added to

[1] The use of the word viewpoint in this paper is not to be confused with the approach introduced by Mullery [27] and further extended by Easterbrook and Nuseibeh [9]. We find the term convenient for describing the different stakeholders that are involved in the RE process and hold to the definition of a viewpoint as a style, an area of concern, a specification, a work plan and a work record [12].

exclude a further case and so on until no stored cases satisfy the rule. Stopping rules, which prevent an incorrect conclusion by providing a null conclusion, are added in the same way. Surprisingly the expert provides a sufficiently precise rule after two or three cases have been seen [19]. Multiple Classification RDR (MCRDR) is defined as the triple <rule,C,S>, where C are the children/exception rules and S are the siblings. All siblings at the first level are evaluated and if true the list of children are evaluated until all children from true parents have been exhausted. The last true rule on each pathway forms the conclusion for the case. Figure 1 shows an example of an MCRDR.

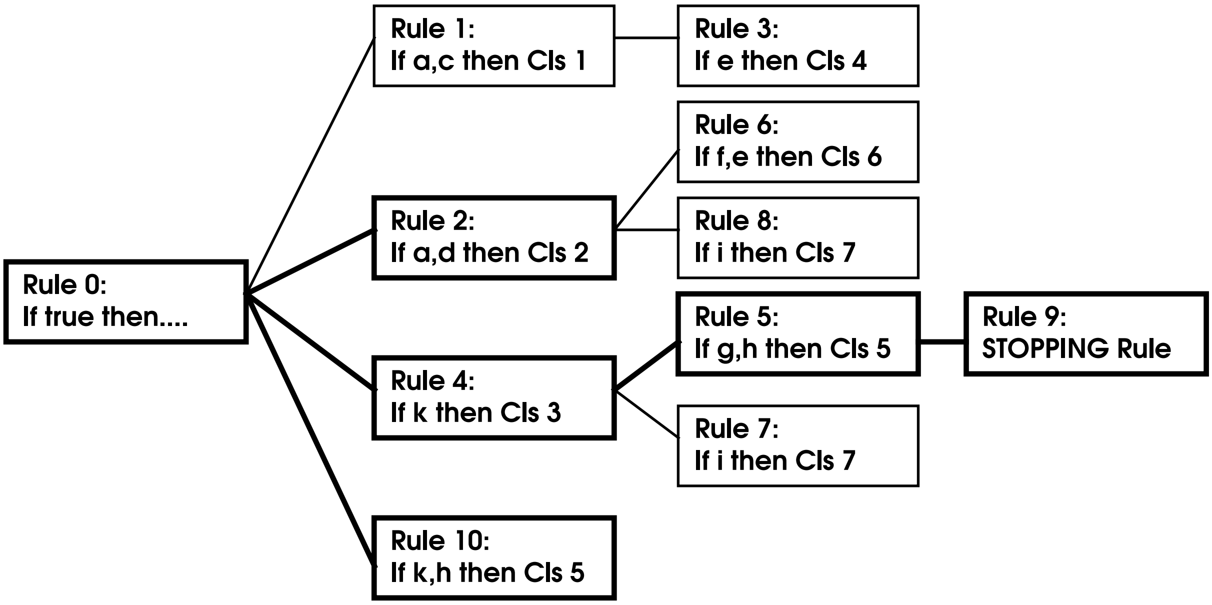

Figure 1. An MCRDR KBS.

The highlighted boxes represent rules that are satisfied for the case {a,d,g,h,k}. We can see that there are two independent conclusions for this case, Class 2 (Rule 2) and Class 5 (Rule 10). Rule 5 had been the cause of a conflict between viewpoints. To resolve this conflict it was decided that attribute g should be dropped. The STOPPING RULE in Rule 9 is used to say that this pathway should not fire, so even though the case satisfies Rule 5 that rule is stopped from being reported. We can see that Rule 10 now replaces the rule pathway for Rule 5 dropping the attribute g.

The success of RDR has been demonstrated in the Pathology Expert Interpretative Reporting System (PEIRS) [10]. PEIRS went into routine use with approximately 200 rules and grew in a four year period (1990-1994) to over 2000 rules. The system was maintained by the expert and the 2000 rules represent a development time of approximately 100 hours. Currently, a commercial version of MCRDR is in use in 12 pathology laboratories. One system has over 6000 rules that were acquired at a rate of one rule per minute. The commercial version continues to be developed and advances in RDR research are being integrated as appropriate.

2.2 Formal Concept Analysis

A concept in FCA is comprised of a set of objects and the set of attributes associated with those objects. The set of objects forms the extent of the concept while the set of attributes forms the intent of the concept. Knowledge is seen as applying in a context and can be formally defined as a crosstable. A formal context ($\mathbb{K}$) has a set of objects G (for *Gegenstande* in German) and set of attributes M (for *Merkmale* in German) which are linked by a binary relation I which indicates that the object g (from the set G) has the attribute m (from the set M) and is defined as: $\mathbb{K}$ = (G,M,I). The following description of FCA follows [45].

A formal concept is a pair (X,Y) where X is the *extent*, the set of objects, and Y is the *intent*, the set of attributes, for the concept. The derivation operators:

$$X \subseteq G: \; X \mapsto X' := \{m \in M \mid gIm \text{ for all } g \in X\}$$
$$Y \subseteq M: \; Y \mapsto Y' := \{g \in G \mid gIm \text{ for all } m \in Y\}$$

are used to construct all formal concepts of a formal context, by finding the pairs (X″,X′) and (Y′,Y″). We can obtain all extents X′ by determining all row-intents {g}′ with g ∈ G and then finding all their intersections. Alternatively Y′ can be obtained by determining all column-extents {m}′ with m ∈ M and then finding all their intersection. This is specified as:

$$X' = \bigcap_{g \in X} \{g\}' \qquad\qquad Y' = \bigcap_{m \in Y} \{m\}'$$

Less formally, we take the set of objects, G, to form the initial extent X which also represents our largest concept. We then process each attribute sequentially in the set M, finding the intersections of the extent for that attribute with all previous extents. Once the extents have been found for all attributes, the intents X′ for each extent X may be found by taking the intersection of the intents for each object within the set. Thereby we determine all formal concepts of the context $\mathbb{K}$ by finding the pairs (X,X′).

To present a visualisation of our ordered set of concepts as a line diagram it is necessary to compute the predecessors and successors of each concept. Predecessors are found by finding the largest subconcept of the intents for each concept. Successors are found by finding the smallest superconcept of the intents. A superconcept is a set that has all of the members of another set and additional members. A subconcept is a set that has fewer members than another set but all the members it has are contained in the other set. We only concentrate on finding sub or super concepts of the intents or extents because they are inversely related and using either set will give the same result. It is interesting to note that the larger the intent the smaller the extent and visa versa. This means that the more attributes we use to describe an object reduces the number of objects that can be described in that manner. Conversely, the more objects in the extent the fewer attributes can be found in the intent which describe those objects. The successor list is used to identify concepts higher in the diagram, the parents, and the predecessor list identifies concepts lower in the diagram, the children. More formally, we use the subsumption relation ≤ on the set of all concepts formed such that $(X_1,Y_1) \leq (X_2,Y_2)$ iff $X_1 \subseteq X_2$. For a family (Xi,Yi) of formal concepts of $\mathbb{K}$ the greatest subconcept, the join, and the smallest superconcept, the meet, are respectively given by:

$$\bigvee_{i \in I} (X_i,B_i) := \left(\left(\bigcup_{i \in I} A_i\right)'', \bigcap_{i \in I} B_i\right) \qquad \bigwedge_{i \in I} (X_i,B_i) := \left(\bigcap_{i \in I} A_i, \left(\bigcup_{i \in I} B_i\right)''\right)$$

From Lattice Theory, the ordered concept set can be used to form a complete lattice, called a concept lattice and denoted B($\mathbb{K}$). The concept lattice structure is seen as a superior representation to a semantic net because it provides “hierarchical conceptual clustering of the objects (via the extents) …. and a representation of all implications between the attributes (via its intents)” (Wille 1992, 497).

3. Building the Framework: The Process Model

The framework is built using a process model made up of the following stages:

1. Requirements acquisition - Capture each viewpoint in an individual propositional knowledge base.
2. Requirements integration - Convert all KBS into a common crosstable format.
3. Concept generation - The crosstable is interpreted as a formal context and FCA is used to derive the concepts associated with each viewpoint.
4. Concept comparison and conflict detection - Pairwise comparison between N views is made to detect conflicts using a four state model of comparison.
5. Negotiation - Here conflict is handled. There are a range of strategies that can be employed based on the types of conflict detected in Stage Four. The resolution operators are employed to update the individual and/or combined views.
6. Evaluation – Determine the degree of conflict to see if viewpoints are moving towards convergence and whether another cycle is needed.

The six steps are iterative. The cycle continues until the parties are satisfied with the combined specification. We will now consider in more detail each of the five phases.

3.1 Phase One: Requirements Acquisition using a knowledge-based approach

There is nothing new about applying knowledge acquisition techniques to the problem of requirements elicitation [7, 18, 25, 32, 39]. A limitation of most earlier approaches is that they only supported a single viewpoint [8]. An exception is Nii [29] who treated the knowledge from each expert as a separate reasoning system. In the approach offered in this paper the goal is to create a shared viewpoint.

This phase does not specify any particular type of knowledge representation or the method of acquisition. However, since phase three and five rely on using FCA, we are only able to make use of knowledge that can be represented as a crosstable. A crosstable is equivalent to a decision table in binary format The decision table is a commonly used representation of knowledge. Further, it has been shown [3] that any decision tree or propositional knowledge base may be converted into a decision table. Thus the restriction of a crosstable representation is still fairly general.

While the framework does not enforce a particular knowledge representation or acquisition technique, MCRDR is proposed due to the issue of requirements change. In RDR maintenance is a simple task that can be performed by the user and does not suffer from the common side-effect problem [41] where a change in one part of the knowledge may invalidate knowledge in other parts of the knowledge base. As discussed in Section 1 of this paper, maintenance is a key concern of the framework offered. RDR is the proposed solution to the handling of evolving and volatile requirements. This paper does not elaborate how RDR handles the maintenance of requirements as space limitations has restricted the focus to requirements comparison. A discussion of the maintenance strategies is offered in [34]. Another reason for the adoption of RDR is the strong element of user involvement as described in section 2.1. It is envisaged that the requirements tool would be used by the user and analyst together. The analyst may be needed to assist the user in understanding the diagrams and in specifying the requirement. It would be up to the group of stakeholders, perhaps under the guidance of an analyst to reconcile conflicts detected by the system and to modify views to bring them into alignment. Unlike many KBS approaches there is no need for a third party (if we see the system and user as one party working together and the analyst as a second party) in the form of a knowledge engineer. User involvement is a critical issue in making RE work [14, 16, 24]. An additional benefit of RDR is that the rule pathways map directly into a decision table and do not need intermediate conclusions to be mapped to primitive conditions [33] as many rule bases require.

SOURCE	INPUT	ACTION	OUTPUT	DESTINATION
Borrower	book	check-in	book	Clerk
	card		card	
Library	book	check-out	book	Borrower

Figure 2: An action table elaborating the agent "Borrower" From [13].

```
1
3
4
1 0 0 0 B1-%BCKIN:(source=borrower) & (input-book) & (output=book) & (action=checkin)
                  & (dest=clerk)
2 0 0 1 B1-%BCOUT:(source=borrower) &(input-card) & (action=checkout) & (output=book)
                  & (output=card) & (dest=borrower)
3 0 0 2 B1-%LCOUT:(source=library) & (input-book) & (action=checkout) & (output=book)
                  & (output=card) & (dest=borrower)
```

Figure 3: The MCRDR rule base for the Borrower Viewpoint. The first three lines show the first rule number, the total number of rules and the next rule number. In this small example there is no default rule and no exception rules. The structure of each rule is rule number, parent rule number, last child rule number, previous sibling rule number, conclusion code and the conjunction of conditions appear after the colon. This representation encodes the requirement as a production rule. For example, rule 1 states that IF the source is a borrower, the input is a book, the action is a ckeckout and the destination is a clerk THEN it is a Borrower Check-in action from the Borrower viewpoint.

In this paper we do not give an example based on requirements acquired using RDR. [34] provides a full example of the framework using data from the SISYPHUS III geology problem [37]. To make the example more relevant to RE, the data are taken from the Library World [13]. In the Library World there is a hierarchy of agents made up of borrowers, staff (which is further subdivided into librarians and clerks), library and catalogue. In this example only the viewpoints of the Borrower, Librarian and Clerk agents are considered as they are "direct" agents which not only supply or receive information but they also process information [13]. The action tables for the Library World are used to directly acquire rules. That is, the rules are not manually acquired using cases. The action table for the Borrower is shown in Figure 2. The corresponding MCRDR knowledge base for the Borrower action table is shown in Figure 3. The conclusion codes (e.g. %BCKIN, %BCOUT and %LCOUT) may seem unnecessary but the notion of a concept in FCA involves an object and its attributes. The rule conditions for the conclusion %LCOUT can be seen as constraints of the system with the conclusion code being the name given to that constraint. If we were to ignore the object name it would be similar to defining a concept such as suckles-young, warm-blooded and vertebrate without naming the object to be a mammal. It is then much more difficult to compare or critique two concepts from different parties as we are not sure to which object the description refers.

To gain the benefits of the RDR approach it is important that a suitable source of cases be available for RE. Apart from the role of cases in RDR, cases offer added benefits for RE. Cases are useful in negotiation as counterexamples for discussion (particularly when associated with particular concepts), they ground the knowledge in the real world, they are generally comprehensible to users and can be used to motivate KA and modification of other views. The way this works is that when a concept is found to be in conflict we pass the case or cases associated with that concept to the other stakeholder for KA. This should either resolve the conflict or at least ensure that both parties have given their views based on the same information. Use cases [35] have gained popularity as a means of eliciting and specifying requirements. We propose use cases as a possible source of cases. Cases used by RDR are often historical cases such as pathology results for a patient. Cases in a case-based reasoning sense are more general and represent stereotypical cases for the domain. Use cases are often developed based on a real situation but then generalised for reusability. Use cases tend to have more detail than would be desirable or feasible in the framework offered. To overcome this the natural language description in the use case can be converted into a more succinct representation such as shown in the diagrams in this paper. It is undesirable that simplification would result in any loss of content. The transition from use cases to cases that are simplified for KA using RDR is an important step that will be investigated in the near future.

3.2 Phase Two: Requirements integration

Requirements integration is the process of ensuring that all viewpoints are in comparable formats. All formats are mapped to the cross table format (to enable the use of FCA in subsequent phases) so that the largest number of mapping schemes we require for N viewpoints (i.e. each viewpoint is captured in a different representation) is 2N not N^2. In the best case, all viewpoints may be captured using the same knowledge representation requiring one mapping scheme. If the requirements are captured directly in cross table format then this phase is not necessary. In our example, the rules shown in Figure 3 are converted into the crosstable shown in Figure 4.

3.3 Stage Three: Concept Generation

In the framework, concepts are generated using FCA. As described in Section 2.2, FCA begins with the definition of a formal context. We interpret the decision table/crosstable in Figure 4 as a formal context where the rows are objects and the columns are attributes. The formal context $\mathbb{K}$ for the Borrower viewpoint in the Library Domain has the set of objects G = {Borrower check-in, Borrower check-out, Library check-out} and the set of attributes M = {source borrower, source library, input book, input card, action check-in, action check-out, output book, output card, destination borrower, destination library}. The crosses show which objects have which attributes, thus I = {(Borrower check-in, source borrower), (Borrower check-in, input book),…,(Library check-out, destination borrower)}. This crosstable is used to find formal concepts as described in section 2.2. The screen dumps shown in Figures 5, 6 and 7 are from an implementation called MCRDR/FCA, which is an enhancement of the current MCRDR for Windows system.

	Source-borrower	source library	input book	input card	action check-in	action check-out	output book	output card	dest. Borrower	dest. clerk
Borrower-check-in	X		X		X		X			X
Borrower-check-out	X			X		X	X	X	X	
Library-check-out		X	X			X	X	X	X	

Figure 4: Context of "Library from Borrower Viewpoint"

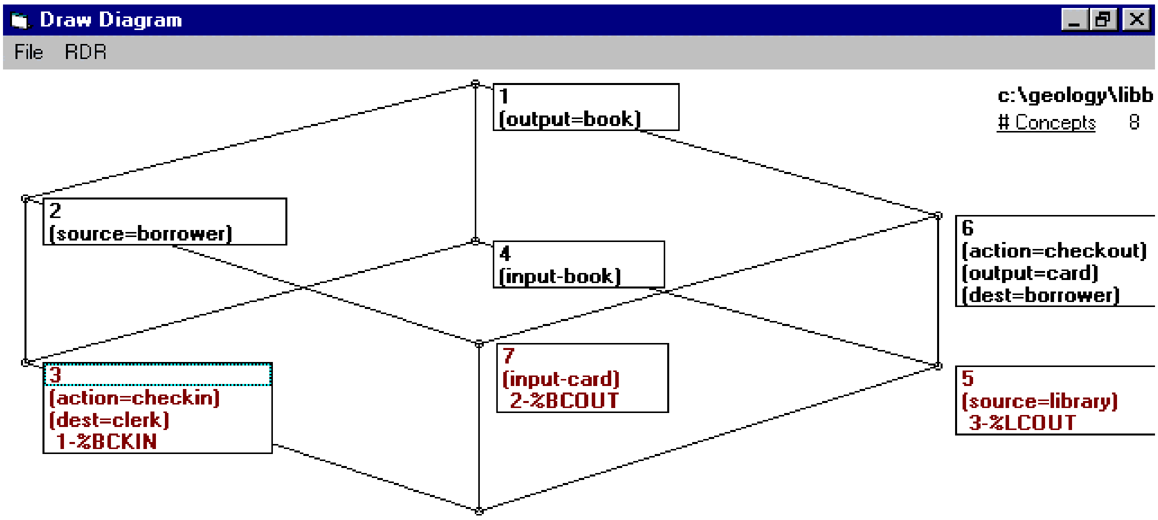

Figure 5: The Line Diagram for the Formal Context "Library from Borrower Viewpoint" in Figure4. Each circle represents a concept where the sets of attributes and objects for that concept are reached by traversing ascending and descending paths,, respectively.

In Figure 5 the concepts are shown as small circles and the sub/superconcept relations as lines. Each concept has various intents and extents associated with it. The labelling has been reduced for clarity. All intents of a concept β are reached by ascending paths from β and all extents are reached by descending paths from the concept β. For example, concept number 7 in Figure 5 contains the set of attributes {source=borrower, action=checkout, output=book, output=card, dest=borrower, input=card} for the object {borrower check out - %BCOUT}. The top node represents the concept that includes all objects and the set of attributes shared by all objects. In Figure 5 all objects share the attribute (output-book). The bottom concept represent the set of objects that share all attributes. In Figure 9, there is no such object. In MCRDR/FCA it is possible to display the concept, attribute/s or object/s belonging to each node or all three dimensions can be displayed concurrently, as in Figure 5. Although the additional labelling clutters the screen the extra information is important in understanding the diagram presented. Alternatively, the user can click on an individual node to see the concept number and all of its extents and intents.

A formal context and line diagram may be developed for each of the viewpoints in the Library World. A line diagram including all viewpoints can be shown to reveal differences between viewpoints. In Figures 6 and 7 we have reduced the context to the checkout action so that the number of concepts being displayed and considered is kept to a manageable number. This is a technique we often use to overcome the limitations of showing large graphs. Alternatives are to allow scrolling and zooming. We prefer to use restricting the focus of attention as this is a common way that humans deal with a large problem. A combination of both techniques is something we want to provide in the future. The process of conflict detection is considered next.

3.4 Stage Four: Concept Comparison and Conflict Detection

The concepts generated for each viewpoint in Phase Three are compared in this step. Different types of conflict can be identified. Easterbrook [8] defines three types: interpretation, design, terminology. CONMAN [36] has six different types of inconsistencies - full, type, version, derivation, link and reachable consistency. The four quadrant model of comparison between experts developed by Shaw and Gaines [38] has been chosen in this study partly because in the future we plan to explore the relevance to our work of a technique they use known as repertory grids, which is based on personal construct psychology [20]. The model of comparison classifies two conceptual models as being in one of four states:

1. Consensus - same concept, same terminology.
2. Correspondence - same concept, different terminology.
3. Conflict - different concepts are being described but the same terms are used.
4. Contrast - there is no similarity between concepts or the terminology used.

States 2, 3 and 4 can all be viewed as conflict states in the more general sense of the word used in this paper. Shaw and Gaines' model, however, does offer greater precision in describing the nature of the conflict which is important in deciding how it can be handled. We will use these four states to describe how the models have been compared. Viewing concepts as being in different states is similar to the work on overlaps [42] where the relation between interpretations of components of two specifications is determined and used in resolving inconsistencies. However, the nature of the overlap is considered in terms of *total*, *partial*, *inclusive* or *non-overlapping*. This does not assist in determining the cause of the overlap which is offered in Gaines and Shaw's four-state model.

Uncovering the consensus between the conceptual models is important for establishing common grounds from which the differences can be viewed. Consensus between viewpoints is found by taking one concept at a time in one viewpoint and making a pairwise comparison with each concept in the other view. More specific to our use of FCA, comparison of two concepts involves testing that the set of attributes and the set of objects for each concept are the same. When a match is found the search is terminated, otherwise the process continues until there are no more concepts. This can be done for any number of viewpoints. If there is a restriction that a concept in consensus must be found in all viewpoints then search of other viewpoints is terminated as soon as a concept is not found in one view. If the goal is to find partial consensus between views, which would include concepts found in at least one other view, then if a concept was not found in one view, the next view would be searched. For concepts in any state except full consensus, it is necessary to repeat duplicate (shared) concepts and annotate the origin of the concept.

Contrast is the opposite situation to consensus where none of the attributes or objects that make up a concept in one viewpoint are found in another viewpoint. This requires pairwise comparison between the concept in question and all concepts in the other view. If any of the attributes or objects in the concept are found in a concept in another view then the search is terminated and the next concept is evaluated for contrast. If more than two viewpoints are being compared then we may get a set of concepts that partially contrast. The line diagram will reveal visually when a concept is in a state of contrast because it cannot be reached in either direction by other concepts.

A concept not in a state of consensus (match found in other viewpoint) or contrast (completely different to all concepts in another viewpoint) is then either in a state of correspondence or conflict. The key to deciding which state it belongs to depends on the terminology. In our approach it would be up to the stakeholder to decide whether terminology used for an attribute or object was the cause for two concepts not appearing at the same node. An alternative approach would be to use the comparison techniques of Gaines and Shaw [38] to identify where terminology is the cause of inconsistency. Shaw (personal communication) has stated that once the differences were pointed out, if the problem was simply terminological it was rectified very quickly and did not require extensive negotiation. Whether we rely on the user or the repertory grid approach to determine if there is a discrepancy in terminology, differences in terminology can be resolved via a number of strategies. In the next section an example is given showing concepts in states of consensus and conflict and what might be done to resolve differences.

3.5 Stage Five: Conflict Negotiation

Before we can determine how to fix a detected inconsistency we need to provide a conflict resolution strategy. There are a number of resolution methods which include negotiation, arbitration, coercion and education [43]. Negotiation is the most appropriate within the assumed context of parties of equal status and ability. An approach to negotiation can be a general, genial chat but since automation is an important goal of requirements engineering research, it is desirable to offer as much automation as possible. Having decided on the nature of the conflict it can be handled in a number of different ways. Thomas [44] offers five main approaches which are: competition, collaboration, avoidance, accommodation and sharing. Easterbrook [8] has described these orientations as domination, integration, neglect, appeasement and compromise, respectively. For Easterbrook and Nuseibeh [9] acting in the presence of inconsistency involves - resolving (remove inconsistency), ignoring (take no action), circumventing (don't include), delaying (put on hold) and ameliorating (reduce the degree of inconsistency). The categories provided by Easterbrook and Nuseibeh appear adequate to the current work. Using this list we now consider how the conflicts identified in the previous phase can be managed.

Contrasting concepts may be handled in a number of ways. Ignoring or cirmcumventing are appropriate tactics if differences are orthogonal, when the concept is not that important or pursuing it is not worth the effort or harm it may cause to the end solution. Delaying is a useful strategy where the issue does not need to be resolved immediately. A policy decision could be made that automatically drops contrasting concepts from the combined viewpoints shown to the stakeholders. However, a safer alternative is to bring contrasting concepts to the user's attention so that they can determine whether the concept should be kept in the view or dropped as irrelevant or unimportant. It could be that one viewpoint is broader than another, that some factors (attributes or objects) have been left out unintentionally or it could be that they were inappropriate. The idea of marking a concept as "ignored" is useful because it identifies that the concept has been considered irrelevant not just overlooked. If it turns out that something has been overlooked then the view in error must be updated appropriately. In MCRDR/FCA it is possible to tag a concept as ignored, circumvented or delayed. Circumvented concepts are not used in generation of the lattice, ignored and delayed concepts are highlighted and shown with the appropriate tag or ignored concepts can be hidden from view.

If the cause of disagreement is different use of terminology then one solution is to update all views to conform to an agreed upon set of terminology. This option is probably not satisfactory to the various stakeholders and also means that some of the history of the specifications is lost. The simple approach used in MCRDR/FCA is to use a synonym or subsumes table which map terms into a common set which are used for comparison. This allows users to keep their own terminology but lets the system reason using a shared set of terms. The mapping step adds to computing overhead and may need to be reconsidered if processing times are unacceptable to the user.

Another situation may be that the set of attributes or objects are partially shared by another concept. To reconcile these concepts it may be decided to drop or add the attributes or objects that differ so that there is consensus. This means that it must be possible to modify viewpoints. This paper takes the view that requirements are never complete and systems are continually evolving. Modification needs to be simple and ensure that changes identified in the concept lattice are performed on the rules in the knowledge base. The combined used of RDR and FCA offers both maintainability and fidelity between the lattice and rules. It is also desirable to be able to temporarily modify the models to support 'what-if' analysis. Those concepts that cannot be reconciled by changes in terminology, tagging or by adding or removing attributes or objects will be those that represent concepts that are left in a state of conflict or contrast.

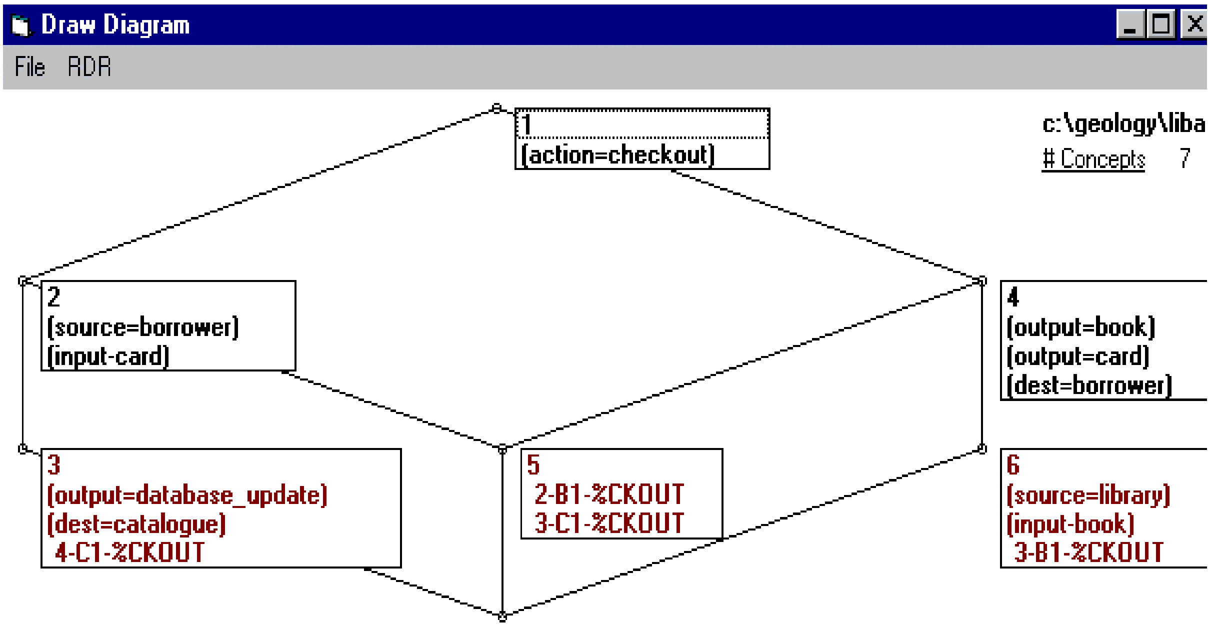

Figure 6: The concept lattice based on the Borrower, Library and Clerk Viewpoints for the checkout action.

In Figure 6 the concept lattice generated using MCRDR/FCA is shown when all concepts related to the checkout action are compared from the Borrower, Library and Clerk viewpoints. There are no correspondence conflicts since the terminology has been defined and restricted according to the action tables provided in [13]. The object is identified by the rule number in the original knowledge base, an identifier for the knowledge base from which it originated (C1 is the Clerk viewpoint, B1 is the Borrower viewpoint and no concepts for the checkout action are contained in the L1- Library viewpoint) and the checkout conclusion %CKOUT. For example, the Borrower and Clerk viewpoints share concept 5 which inherits the attributes action=checkout (concept 1), input=card, source=borrower (concept 2), output=book, output=card, dest=borrower (concept 4). However, the borrower views the library as an input source of books (concept 6) which is not shared by the clerk due to their different perspectives. The Clerk's viewpoint includes updating the catalogue database (concept 3). As would be expected, this concept is not shared with the borrower. Thus Figure 5 includes concepts in consensus and concepts in conflict, but the inconsistencies are acceptable due to the different roles each viewpoint plays. However, if the borrower had believed that the library was the input source for a card, we would have the concept lattice shown in Figure 7, which would reveal an inconsistency that needed to be addressed (compare the set of attributes for concepts 7 and 9). The increased complexity of the lattice is noteworthy, there are now 11 concepts instead of the 7 concepts in Figure 6 and no concepts are shared by the Borrower and Clerk viewpoints. The cause of this conflict (misunderstanding) may be that the borrower is thinking in terms of the originating source of the card or the borrower did not realise they had to bring their card with them each time.

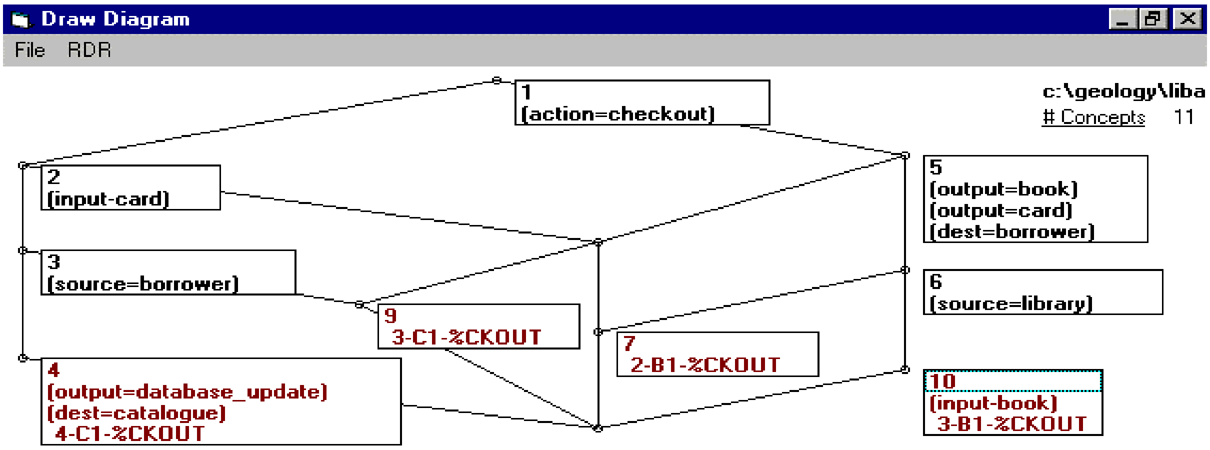

Figure 7: The concept lattice from Figure 6 when the Borrower viewpoint is changed to define the library as the input source of a card, ie, source=borrower is changed to source=library..

3.5.1 Living with inconsistency

There are a number of researchers who accept that conflict resolution cannot always be negotiated. These approaches can be termed as living with inconsistency or 'lazy' consistency [28] and can be compared to fault-tolerant systems that continue to function after non-critical failures occur [31]. Finkelstein et al [13] don't want to remove all conflict because forced removal "constrains the specification unnecessarily " (p. 2) and "tends to restrict the development process and stifle novelty and invention" (p. 4). They see that consistency is necessary within a viewpoint but partial consistency between viewpoints is allowable. Similar to our use of tags, another approach is to use "pollution markers" [1] that act as a warning that code may be unstable or that the users should carefully check the output. They can be used to screen inconsistent data from critical paths that must have completely consistent input. [13] claim that their approach goes further by stating how inconsistency should be acted upon by using rules which describe how different inconsistencies should be handled. Amelioration allows the degree of conflict to be reduced. If we think in terms of the concept lattice we would be shortening the number of edges we need to traverse to get from one concept to the other.

3.6 Stage Six: Evaluation

The purpose of this stage is to evaluate if the distance between viewpoints is decreasing and that we are moving towards a shared viewpoint from which the specifications can be developed. Using the lattice we could use techniques from graph theory to compute the distance. A very simple measure that can be used when comparing the distance between the same concept (i.e. they share the same object/conclusion) in each viewpoint is to compute the sum of the number of attributes which are shared, by the total number of attributes for that concept in the viewpoints being compared. Concepts in contrast (no match) get a score of 0, concepts in consensus (complete match) get a score of 1. These scores can be combined to produce an overall measure of similarity. The higher the score the more similar are the viewpoints. We can use this measure to determine if we are reaching a shared viewpoint and whether another iteration is necessary. A related issue is evaluation of the completeness of the set of requirements. The answer to this is that requirements are never complete although it is hoped that by combining requirements from a number of viewpoints and by repeating the six phase process until a suitable state of (partial) consensus is reached that there are minimal gaps in the final requirements model.

4 Future Work and Conclusion

This work seeks to address two problems that are shared by KE and RE research: handling change in knowledge/requirements and dealing with multiple sources of expertise/viewpoints. This paper has focused in more detail on the latter problem and suggested the use of RDR for the former problem. In addressing the issue of multiple viewpoints, a framework has been presented which captures viewpoints into individual KBS, develops conceptual models of each KBS which are compared and reconciled using a number of resolution strategies. The strategies include: modifying the viewpoints, showing cases related to a concept, the use of tags to mark concepts, the ability to visualise what concepts are contained in each viewpoint and the measurement of distance between viewpoints. The example offered from the library domain bypassed the acquisition phase as the requirements had already been acquired. A key assumption of the framework is the ability to represent requirements as a formal context. More specifically, to address the maintenance issue, RDR are proposed as the preferred KA technique. RDR have already found commercial success which will be enhanced by current research into conceptual modelling [33], fuzzy reasoning [26], the application of RDR to more problem types and the extension of RDR beyond propositional rules to first order logic. The most pressing issue for the use of RDR in the framework proposed is whether suitable cases can be made available for KA. At this point, use cases appear to be the most viable option for capturing initial requirements in a case format. Exploring this idea is the next step.

The system described offers automation of Phases Two and Three which concern representation of the knowledge and generation of concepts. Phases One, Three and Five of acquisition/specification, conflict identification and negotiation, respectively, are partially automated as they are seen to be essentially human activities. A good solution to a user’s problem will require creativity and creativity is not something that can be automated [8]. In a similar vein, the framework offered in this paper was designed to support users by assisting them to develop both individual and group requirements viewpoints.

Aknowledgements

Thanks to Tim Menzies who suggested that the MCRDR/FCA approach could be applied to the problem of requirements engineering and who participated in a number of useful discussions to this end.

References

[1] Balzer, R. (1991) Tolerating Inconsistency *Proceedings of 13th International Conference on Software Engineering (ICSE-13)* Austin, Texas, USA, 13-17th May 1991, IEEE Computer Society Press, 158-165.
[2] Clancey, W.J., (1993) Situated Action: A Neurological Interpretation Response to Vera and Simon *Cognitive Science*, 17: 87-116.
[3] Colomb, Robert.M. (1993) Decision Tables, Decision Trees and Cases: Propositional Knowledge-Based Systems *Technical Report No. 266* Key Centre for Software Technology, Department of Compter Science, The University of Queensland, Australia.
[4] Compton, P. and Jansen, R., (1990) A Philosophical Basis for Knowledge Acquisition. *Knowledge Acquisition* 2:241-257.
[5] Compton, P., Edwards, G., Kang, B., Lazarus, L., Malor, R., Menzies, T., Preston, P., Srinivasan, A. and Sammut, C. (1991) Ripple Down Rules: Possibilities and Limitations *6th Banff AAAI Knowledge Acquisition for Knowledge Based Systems Workshop*, Banff, 6.1 - 6.18.
[6] Dobson, J.E., Blyth, A.J.C., Chudge, J. and Strens, M.R. (1994) The ORDIT Approach to Organisational Requirements In Jirotka, M. and Goguen, J. (eds) *Requirements Engineering: Social and Technical Issues* Academic Press, London, 87-106.
[7] Easterbrook, S. M. (1989) Distributed Knowledge Acquisition as a Model for Requirements Elicitation, In *Proceedings of the 3rd European Workshop on Knowledge Acquisition for Knowledge Based Systems*, Paris, July 1989.
[8] Easterbrook, Steve (1991) Elicitation of Requirements from Multiple Perspectives *PhD Thesis*, Department of Computing, Imperial College of Science, Technology and Medicine, University of London, London SW7 2BZ.
[9] Easterbrook, S. and Nuseibeh, B. (1996) Using Viewpoints for Inconsistency Management *BCSEEE Software Engineering Journal* January 1996, 31-43.
[10] Edwards, G., Compton, P., Malor, R, Srinivasan, A. and Lazarus, L. (1993) PEIRS: a Pathologist Maintained Expert System for the Interpretation of Chemical Pathology Reports *Pathology 25*: 27-34.
[11] Edwards, G., Kang, B., Preston, P. and Compton, P. (1995) Prudent Expert Systems with Credentials: Managing the expertise of Decision Support Systems *Int. Journal Biomedical Computing* 40:125-132.
[12] Finkelstein, A.C.W., Goedicke, M., Kramer, J. and Niskier, C. (1989) Viewpoint Oriented Software Development: Methods and Viewpoints in Requirements Engineering In *Proceedings of the Second Meteor Workshop on Methods for Formal Specification* Springer Verlag, LNCS.
[13] Finkelstein, A., Gabbay, D., Hunter, A, Kramer, J. and Nuseibeh, B. (1994) Inconsistency Handling in Multi-Perspective Specifications *IEEE Transactions on Software Engineering* 20(8):569-578.
[14] Gause, D. D. and Weinberg, G. M. (1989) *Exploring Requirement: Quality Before Design* Dorset House Publishing, New York.
[15] Greenspan, S. Mylopoulos, J. and Borgida, A. (1994) On Formal Requirements Modelling Languages: RML Revisited *Proceedings of 16th International Conference on Software Engineering*, IEEE Press, New York, 135-148.
[16] Herlea, D.E. (1996) User's Involvement in the Requirements Engineering Process *Proceedings 10th Knowledge Acquisition Workshop*, Banff, Canada.
[17] Ignizio, James, P. (1991) *Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems* McGraw-Hill Inc, New York.
[18] Johnson, W. L., Feather, M. S. and Harris, D. R. (1992) Representation and presentation of Requirements Knowledge, *IEEE Transactions on Software Engineering*, 18(10): 853-869.
[19] Kang, B., Compton, P. and Preston, P. (1995) Multiple Classification Ripple Down Rules: Evaluation and Possibilities *Proceedings 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop* Banff. Feb 26 - March 3 1995, Vol 1: 17.1-17.20.
[20] Kelly, G.A, (1955) *The Psychology of Personal Constructs* Norton, New York.
[21] Lam, W., Shankararaman, V. and Jones, S. (1988) Managing Requirements Change: A Set of Good Practices E. Dubois, A. L. Opdahl and K. Pohl (eds) *Proceedings of the Fourth International Workshop on Requirements Engineering: Foundation of Software Quality, (REFSQ'98)* Pisa, Italy 8-9 June, 1998. Presses Universitaires De Namur, 85-97.
[22] Lane, M. and Cavaye, A.L.M (1998) Management of requirements volatility enhances software development productivity In *Proceeding of the 3rd Australian Conference on Requirements Engineering (ACRE98)*, Geelong, Australia, 49-63.
[23] Larkin, Jill H. and Simon, Herbert, A. (1987) Why a Diagram is (Sometimes) Worth Ten Thousand Words Cognitive Science 11:65-99.
[24] Macaulay, L. A. (1996) *Requirements Engineering* Springer Verlag, Berlin.
[25] Maiden, N. A. M. and Rugg, G. (1994) Knowledge Acquisition Techniques for Requirements Engineering *In Proceedings of the Workshop on Requirements Elicitation for Systems Specification*, Keele, UK, July 1994.
[26] Martinez-Bejar R., Shiraz G. M., and Compton P. (1998) Using Ripple Down Rules-based Systems for Acquiring Fuzzy Domain Knowledge, In Gaines, B. and Musen, M. (eds) *Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW'98),* Banff, Alberta, Canada, 18-23 April, 1998, Vol 1:KAT.2.
[27] Mullery, G. P. (1979) CORE - a method for controlled requirements expression *In Proceedings of the 4th International Conference on Software Engineering (ICSE-4)*, IEEE Computer Society Press, 126-135.

[28] Narayanaswamy, K. and Goldman, N. (1992) "Lazy Consistency": A Basis for Cooperative Software Development *Proceedings of International Conference on Computer-Supported Cooperative Work (CSCW'92)* Toronto, Ontario, Canada, 31 October- 4 November, ACM SIGCHI & SIGOIS, 257-264.

[29] Nii, H.P. (1986) Blackboard System (Part 2): Blackboard Application Systems, Blackboard Systems from Knowledge Engineering Perspective *AI Magazine* Summer 1986.

[30] Norman, D.A.(1986) Cognitive Engineering In Norman, D.A. and Draper, S.W. (eds.) *User Centered System Design: New Perspectives on Human-Computer* Interaction Lawrence Erlbaum Associates.

[31] Nuseibeh, B. (1996) To Be And Not To Be: On Managing Inconsistency in Software Development, *Proceedings of 8th International Workshop on Software Specification and Design (IWSSD-8),*Scloss Velen, Germany, 22-23 March 1996, IEEE CS Press, 164-169.

[32] Reubinstein, H.B. and Waters, R.C. (1989) The Requirements Apprentice: An Initial Scenario *Proceedings of the fifth IEEE International Workshop on Software Specification and Design* Pittsburgh, Penn.

[33] Richards, D. and Compton, P. (1997) Uncovering the Conceptual Models in Ripple Down Rules In Dickson Lukose, Harry Delugach, Marry Keeler, Leroy Searle, and John F. Sowa, (eds) (1997), *Conceptual Structures: Fulfilling Peirce's Dream, Proceedings of the Fifth International Conference on Conceptual Structures (ICCS'97),* August 3 - 8, University of Washington, Seattle, USA, Lecture Notes in Artificial Intelligence, Number 1257, Springer Verlag, Berlin, 198-212.

[34] Richards, D. and Menzies, T. (1998) Extending the SISYPHUS III Experiment from a Knowledge Engineering to a Requirements Engineering Task In Gaines, B. and Musen, M. (eds) *11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98),* Banff, Canada, SRDG Publications, Departments of Computer Science, University of Calgary, Calgary, Canada, Vol 1:SIS-6.

[35] Rumbaugh, James (1994) Getting Started: Using Use Cases to Capture Requirements *JOOP* September 1994, 8-23.

[36] Schwanke, R.W. and Kaiser, G.E. (1988) Living with Inconsistency in Large Systems *Proceedings of the International Workshop on Software Version and Configuration Control* Grassau, Germany, 27-29 January 1988, B.G. Teubner, Stuttgart, 98-118.

[37] Shadbolt, N., (1996) URL:http://www.psyc.nott.ac.uk/aigr/research/ka/SisIII

[38] Shaw, M.L.G. and Gaines, B.R., (1988) A Methodology for Recognising Consensus, Correspondence, Conflict and Contrast in a Knowledge Acquisition System in *Proceedings of the 3rd Knowledge Acquisition for Knowledge-Based Systems Workshop,* November 1988, Banff, Canada.

[39] Shaw, M. L. G. and Gaines, B. R. (1996) Requirements Acquisition *Software Engineering Journal*, May 1996, 149-165.

[40] Shaw, M.L.G. and Woodward, J.B. (1989) Mental Models in the Knowledge Acquisition Process, In *Proceedings of Fourth Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banff, October 1989.

[41] Soloway, E, Bachant, J. and Jensen, K. (1987) Assessing the Maintainability of XCON-in-RIME: Coping with Problems of a very Large Rule Base *Proceedings of the Sixth International Conference on Artificial Intelligence*, Seattle, WA: Morgan Kaufman, Vol 2:824-829.

[42] Spanoudakis, G. Finkeltein, A. and Till, D. (1999) Overlaps in Requirements Specifications Automated Software Engineering Journal (to appear) available from ftp.cs.city.ac.uk/users/gespan/ase99.ps.gz

[43] Strauss, A. (1978) *Negotiation: Varieties, Contexts, Processes and Social Order* Jossey-Bass Publishers, San Francisco, CA.

[44] Thomas, K. (1976) Conflict and Conflict Management In Duneette (ed) *Handbook of Industrial and Organisational Psychology* Rand McNally College Publishing Co.

[45] Wille, R. (1982) Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts In I. Rival (ed), *Ordered Sets* Reidel, Dordrecht, Boston, 445-470,.

[46] Zowghi, S. and Nurmuliani (1998) Investigating requirements volatility during software development: Research in progress In *Proceeding of the 3rd Australian Conference on Requirements Engineering (ACRE98),* Geelong, Australia, 38-48.